EFFECTIVE TODAY ~ ESSENTIAL TOMORROW

SIAMEASE SCRIPTING

Simmonds and Associates

Copyright © 2001

Scripting and the Management thereof

Functional User Manual

SIAMEASE 3.0

SAP R/3 Automation

SIMMONDS AND ASSOCIATES

COPYRIGHT © 2001

CONTENTS

Chapter 1 FOREWORD 4

Chapter 2 DOCUMENT SPECIFICATIONS 4

Chapter 3 METHODOLOGY 5

Chapter 4 SIAMEASE SCRIPTS 6

OVERVIEW 6

Chapter 5 SIAMEASE SCRIPT BUILDER ® 7

OVERVIEW 7

CREATE A SCRIPT 8

STEPS 9

ADDING AND REMOVING TASKS 10

REPEATING STEPS 11

NAVIGATION 11

ADDING ATTACHMENTS 12

DEFAULT ATTACHMENT PATH 12

ADDING MEMO TEXT 13

Chapter 6 SCRIPT RULES 14

SCRIPT VARIABLES 15

VARIABLES 15
COUNTER 15
STEP 15
ITEM & LINE 15

OUTBOUND RULES 16

INBOUND RULE 19

ENHANCED VARIANT CONSIDERATIONS 24

ADDING SIAMEASE FORMULAE TO SPREADSHEET DATA 24
VARIANT FORMULA 24

LINKING SCRIPTS 25

SCRIPT RULE SUMMARY 26

Chapter 7 SIAMEASE SCRIPT MANAGER ® 27

OVERVIEW 27

CREATING AND CHANGING SCRIPTS 28

CREATING NEW SCRIPTS 28
CHANGING EXISTING SCRIPT 28

EXECUTING SCRIPTS 28

EXECUTE SCRIPT 28
SELECTING TASKS / SCRIPTS 30
RUN PARAMETERS 30

VIEW RUN RESULTS 31

Chapter 8 APPENDIX A – CASE STUDIES 33

THE TRAINING CLIENT 33

SCENARIO TESTING 34

STRESS TESTING 34

Chapter 1

FOREWORD

Special thanks to all the people who have helped with the development and testing of Siamease. Many specialist consultants have volunteered their business and technical advice to make Siamease a product that most SAP projects cannot be without.

Siamease provides a wide range of functionality. It does this by exploiting SAP automation, thus allowing users to execute SAP transactions in bulk using data that has either been uploaded from a legacy system or data that already exists within the customer SAP database. Users are therefore able to perform data migration, build interfaces, maintain live customer data, build training systems and build test scripts.

This document explains all options available within Siamease. Separate documents exist that walk through business scenarios. Siamease consists of 4 components, a Task Builder, a Task Manager, a Script Builder and a Script Manager.

Please feel free to email your comments to us at simmonds@global.co.za

Chapter 2

DOCUMENT SPECIFICATIONS

Prepared by:	Simmonds and Associates
Released:	2001-04-27
SAP Versions:	4.6
Audience:	SAP Implementation Teams, Super Users, Development team, Functional Consultants, Data Administrators, Training Teams, Basis stress testers, Project Managers.
Trademarks	Microsoft Excel ® is a trademark of Microsoft.

Chapter 3

METHODOLOGY

Siamease is based on a methodology that empowers the business. It does this by bridging the gap between the functional and the technical environments. Most technical development starts with a functional specification, which indicates that if we shorten this process or eliminate its need altogether then we are on the right track. Siamease provides a comprehensive and complete solution to functional team members and does all the technical stuff for them, leaving them to apply business decisions in their area of expertise in respect of data migration, interfacing, maintenance within the system, training and testing.

Siamease helps to break complex problems down to smaller manageable units. Each unit is known as a Task within Siamease and is created using the Siamease Task Builder and managed and controlled via the Siamease Task Manager.

The Siamease Scripting tool allows these smaller components or tasks to be chained together to form complete comprehensive solutions. Each script is built via the Siamease Script builder and managed and controlled by the Siamease Script Manager.

Where a Siamease Task focuses on individual transactions, Scripts focus on processes and building master data by applying several Tasks.

The complete methodology is monitored and controlled via Managers that allow visibility of Tasks and Scripts as well as providing a repository or workplace where scenario data and documentation can be linked to the Tasks at hand. In a live environment the Managers provide users with the workplace where Tasks and Scripts can be executed and results stored and monitored.

Chapter 4

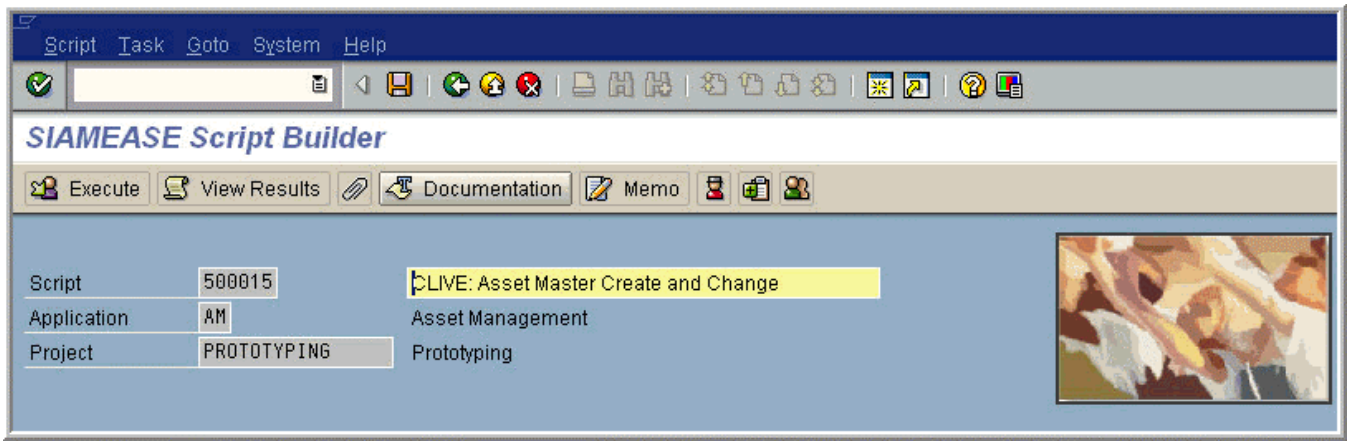
SIAMEASE SCRIPTS

OVERVIEW

Using the Siamease Task builder and Task manager, Tasks are created. Each Task representing a specific business operation performed using a SAP transaction. Script Manager extends Siamease functionality by allowing Tasks to be linked or chained together to process a business scenario. Business Scenarios consists of several transactions that follow each other.

Scripting primarily allows data to flow from one transaction to another. With its intelligent data handling it eliminates the need to build lookup tables in the middle of a business process or data migration. Scripting also focuses on the activities around the business process, thus acting as a repository for documentation and data that can be used within the process.

Refer to Scripting Case studies to see how scripting is used creating training clients, scenario testing and data migration.

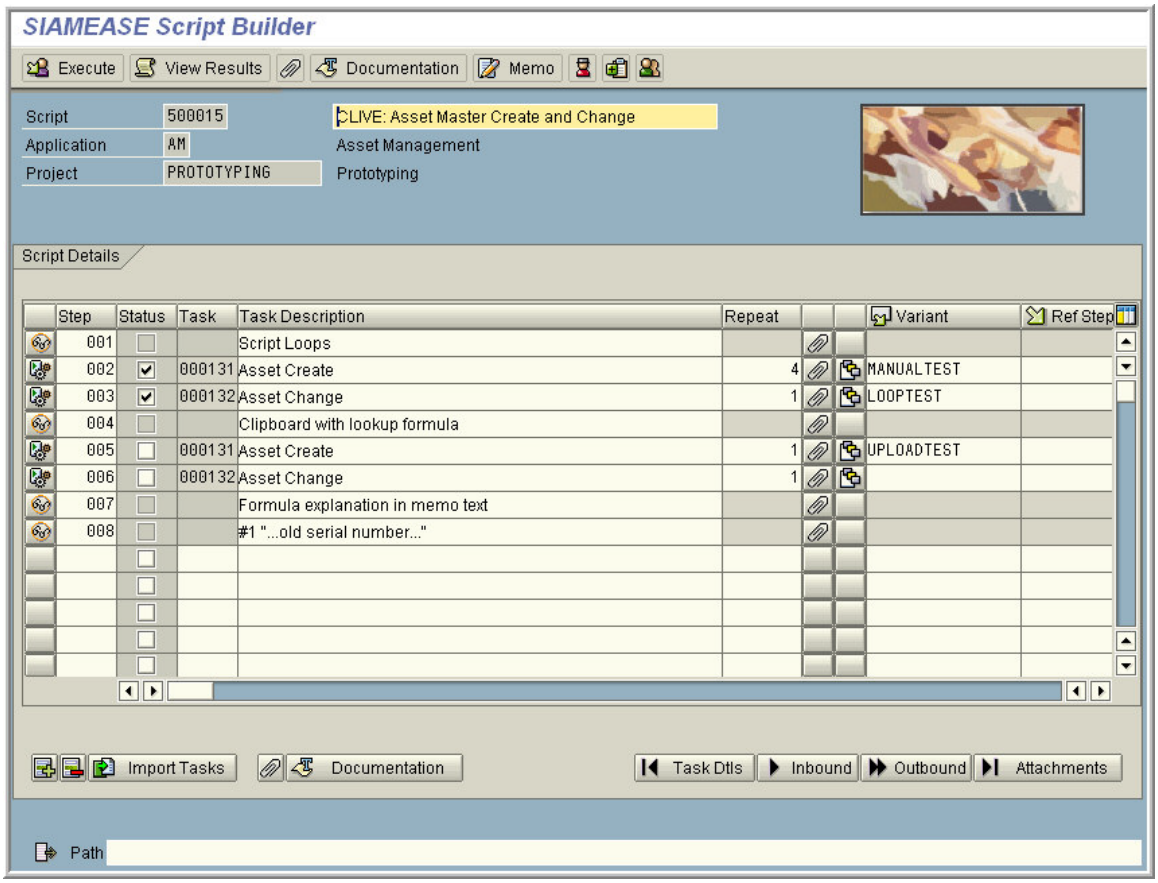


Chapter 5

SIAMEASE SCRIPT BUILDER ®

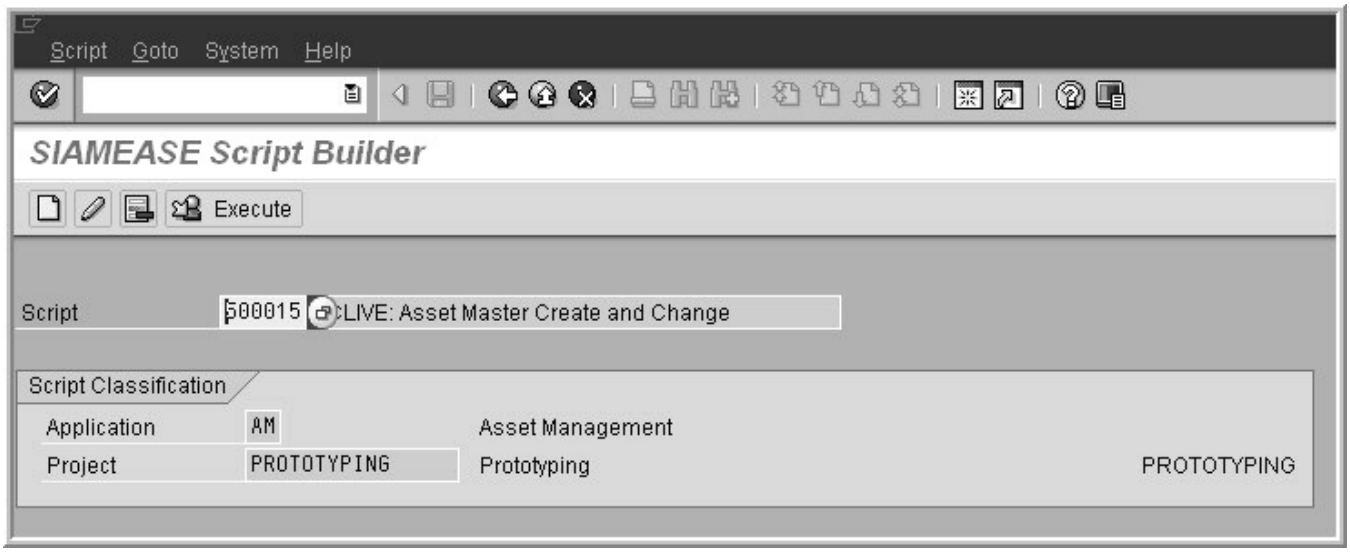
OVERVIEW

Scripts are created with unique document / script numbers and are classified under projects and application areas very much like tasks are. A script is built using the Script Builder and managed via the Script Manager that makes all scripts visible to users and tracks results created by each scripts. Scripting allows scenarios to be formulated and documented. It allows a Functional Team to create an electronic procedure that can be executed to either create scenario data or to test a business scenario. If executed more than once the same procedure can be used to measure system performance in a stress test. In order to simplify certain data loads, several tasks are created. Each however performs a specific task if supplied with the right data to capture. Scripting fundamentally allows data created by one task to be available for input in subsequent tasks. This allows processes to be built as well as allowing problems that have been broken down to be built back to the complete cohesive level.



CREATE A SCRIPT

- Execute transaction ZESB or select the script builder from any Siamease screen.
- Enter a Script description.
- Assign the script to a Functional Area or Application
- Assign the script to a Project.
- Press the CREATE icon.



STEPS

A Task represents a business operation performed on a SAP transaction. Tasks added into a script are known as STEPS. Steps influence the data that is to be used in that operation as well as the sequence of events within a business process. Each step has rules defined to determine what data is passed to a task and what data is stored as a result of a task.

Each step is numbered and can have data or documentation attached to it in the form of an attachment for later reference.

A Step can be

A comment used to document the process

A reference script that executes other scripts

A task imported from the task manager

	Step	Status	Task	Task Description	Repeat			Variant
	001	☐		Script Loops				
	002	☐		SCRIPT				500012
	003	☑	000131	Asset Create	4			MANUALTEST

ADDING AND REMOVING TASKS

Adding a task creates a unique step number.

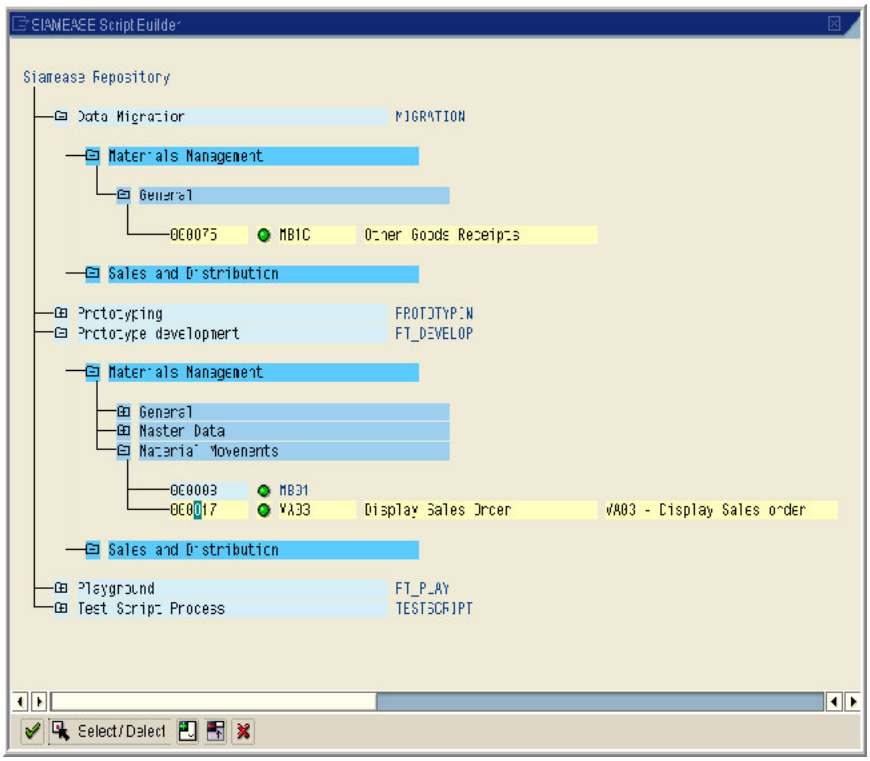
Enter the desired task number into the task column to create a new step.

Click on the icon next to the step number to insert a step between two steps.

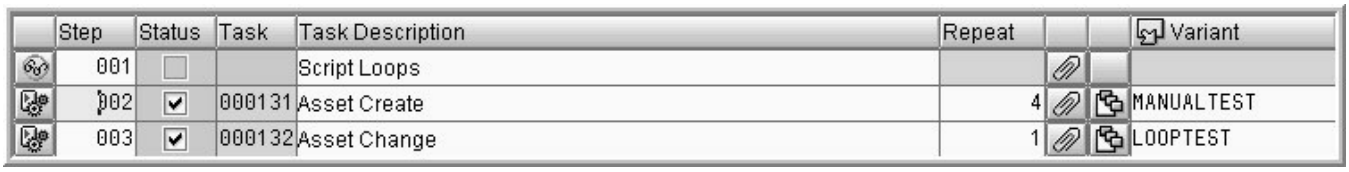
Adding and removing steps can also achieved by pressing the + and – buttons on the bottom left of the script builder screen.

It is suggested to press the IMPORT Tasks button. Siamease will then present all available Tasks for selection. To select, click on the relevant Task and press F9 or the select/deselect icon on the bottom of the screen.

The selected task will be highlighted in yellow.



Once complete press the green-tick icon to continue. The selected tasks will be copied back to the script and presented as follows.



Step	Status	Task	Task Description	Repeat	Variant
001	☐		Script Loops		
002	☑	000131	Asset Create	4	MANUALTEST
003	☑	000132	Asset Change	1	LOOPTEST

- Step numbers can be manipulated to allow tasks to be reordered. Change the value of Step, press Enter and Siamease will reorder the tasks accordingly.
- The status checkbox indicates whether this task will run if the script is executed.
- Task details are
- All steps should run with a variant, unless all selection screens are made visible during the execution.
- Documentation or data can be attached to each step via the attachment icon.

REPEATING STEPS

A step can be executed more than once as well as the overall script being executed more than once. Referring to the REPEAT column above, this value will execute step 3 4 times. Each iteration increments counter variables. Each time a step is repeated it increments the ITEM counter (&I). All results from each step are also separated by this variable. Refer to Script Variables for more details.

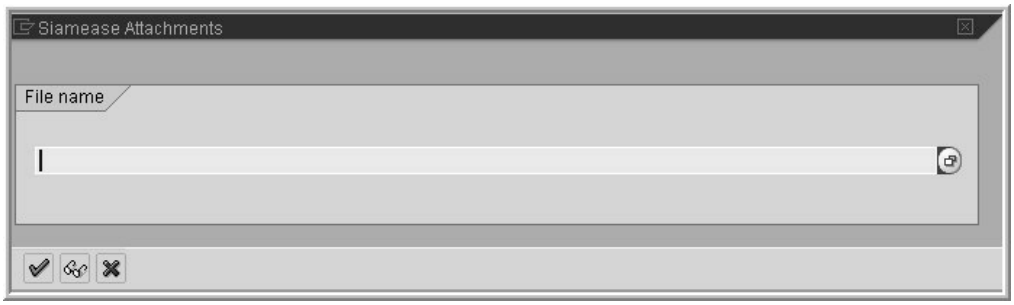
NAVIGATION

These buttons simply allow the user to move to the appropriate columns in the script for easier navigation around the script.

ADDING ATTACHMENTS

Each step can have an attachment linked to it. This allows documentation, presentation material or scenario data to be linked to a specific task.

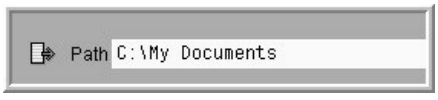
If a step has no attachment then a paper clip icon will appear on the step line. Press this icon to add a hyperlink to the document in question.



Press F4 to find the attachment using the normal windows file explorer.

DEFAULT ATTACHMENT PATH

Each script has a default path from which each attachment can be referenced. The script builder will take the path of the first attachment as the default script path. All other attachments are then compared to the first to determine if the same path is used. If so then only the relative position to the script path is stored. The default script path is visible and can be captured manually in the PATH field on the bottom of the script builder detail screen.

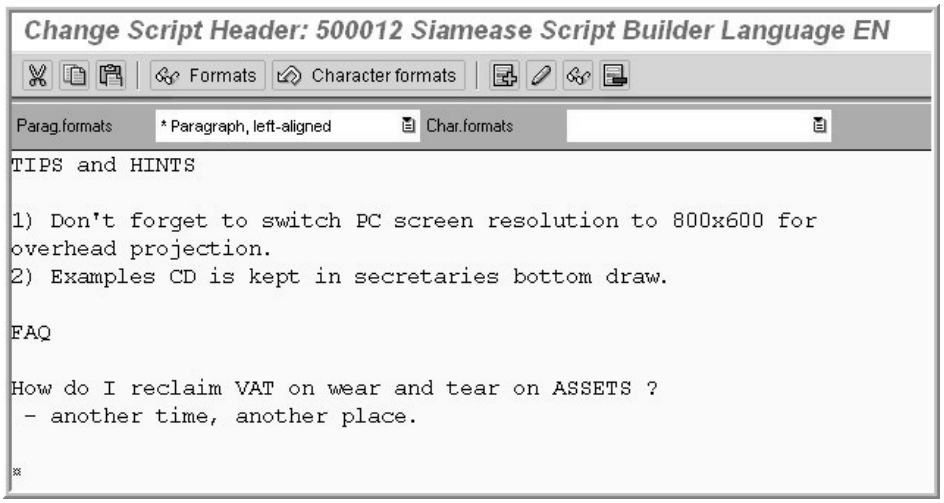


In the example above the second step has an attachment that refers to
`C:\My Documents\testsd2.txt`

ADDING MEMO TEXT

Scripts are rerun over time. The Memo text allows empirical data to be stored alongside the script. The Memo text can even store scenario data that can be copied into the clipboard and imported into the scenario.

To add memo-text press the MEMO icon on the application toolbar.


Change Script Header: 500012 Siamease Script Builder Language EN
Formats
Character formats
Parag.formats
* Paragraph, left-aligned
Char.formats
TIPS and HINTS
1) Don't forget to switch PC screen resolution to 800x600 for overhead projection.
2) Examples CD is kept in secretaries bottom draw.
FAQ
How do I reclaim VAT on wear and tear on ASSETS ?
- another time, another place.

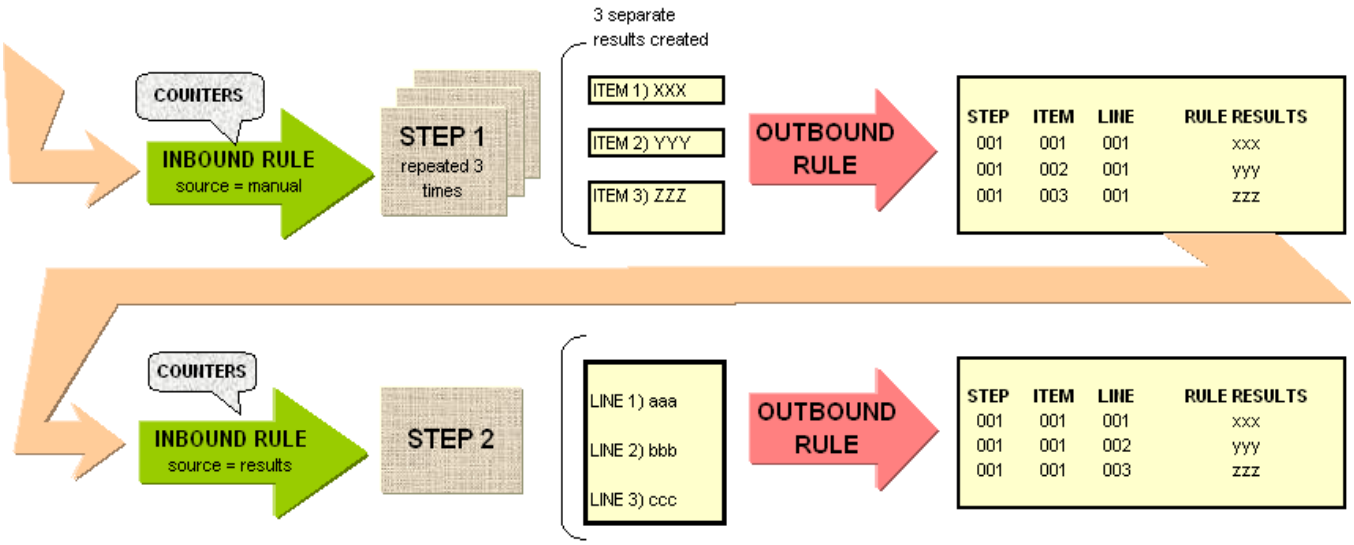
Chapter 6

SCRIPT RULES

Scripts contain rules that determine what data is required for subsequent processing, thus allowing a script to prepare data at runtime to be used for subsequent steps.

Since each Task can load data into SAP from different data sources (e.g. clipboard, manual, memory etc.), the script step must specify the source of data when the step runs. This is done by creating a variant on the task to be executed.

Each step has inbound and outbound rules.



Scripting provides

- a set of variables (counter) which are used by the inbound and outbound rules
- enhanced variant commands that extend SAP functionality to define inbound rules
- outbound rules that allows specific data created by the task to be saved for later use
- dynamic table lookups to assist with data migration and helps to tie legacy data to SAP data

To define scripting rules, script variables will be explained, and then the concept of the outbound rules will be explained, leading into the inbound rules. Scripting also assumes that the user is familiar with the Task and its capabilities.

SCRIPT VARIABLES

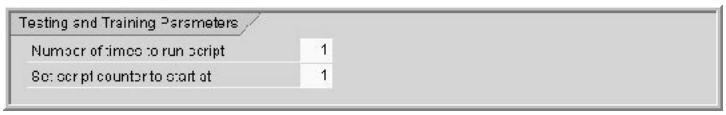
Scripting introduces 4 variables that are automatically incremented when a script is executed. These variables are used to store results and to change field values at runtime.

VARIABLES

- **C** COUNTER - is incremented for each iteration of a complete script
- **S** STEP - the current step that is being run.
- **I** ITEM - the number of times that this step has run
- **L** LINE - the number of transactions that have been created by this item

COUNTER

The Siamease script executor can execute a script a number of times. The C counter variable will increment for each iteration of the entire script. The script executor can also specify a starting point for this counter.



STEP

A Script defines which tasks are to be run. Each task is assigned a step number, which determines the order in which the tasks are run. The execution of a step will set this counter to the step number. If the first task that runs is in step 3 (i.e. the first 2 steps are comments) then this counter will be set to 3, even though it is the first step to run. The reason for this is to allow easy reference to the data created by that step at a later stage.

ITEM & LINE

ITEM will increment each time a specific step is called. The script can specify that a step must be repeated 4 times. This time the step is executed the ITEM counter will increment and the LINE counter will reset to 1. If a specific step were to run with 4 input records, it would then increment the LINE counter for each transaction posted to SAP. If a step were to load data from a spreadsheet then each row loaded from the spreadsheet would be referenced by a LINE value. Item would remain constant since the step was only called once.

It is possible for a step to be repeated and for it to process multiple lines of data. In this case both variables would be changing. These counters reset for each step and run / item.

OUTBOUND RULES

The outbound rules are specified in the script builder. Scroll the step table right to view the rule fields.

The outbound rules determine what data is to be saved after a transaction is posted into SAP. This allows subsequent steps to utilise that data to continue the business process. Each step has 6 rule fields that are configurable. The reason for storing results would be to either report the results of a data load or to use the results later in the script.

Step	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Xref Rule
001						
002	%AN1	.	.	.	.	@1
003	%AN1	$SY-MSGV2	.	.	.	.
004						

RESULTS TABLE

After Siamease posts a transaction into SAP it interprets the steps rules and stores the result using the current script variables. Saved data is stored in a matrix or table called RESULTS, which is then available as a data source in subsequent steps. The RESULTS table is available at the end of the script run for interrogation and can be exported to Excel™ with a single button click.

Note that the results table is only available as a data source from within a script. Results however can be saved after a run. Results are stored in table ZCSRESULTS and can be referenced by a task. In this case the task must be defined upfront to access the SAP table ZCSRESULTS when generating the task.

OUTBOUND VARIANT COMMANDS

The objective of the outbound variant commands is to determine what document or object number has just been created. SAP usually sets a system variable or a Parameter ID to the value of the document or object that has just been created. The Outbound variant commands allow access to either of the two and are discussed in detail below.

%	Refers to a Parameter ID
@	Inbound / input data after conversions can be stored along with new result data
$	A System variable or any other runtime variable

PARAMETER ID % %pid

Most key fields within SAP have a parameter id assigned to them. This parameter id stores the last document or number that was used or created to assist users and programs with the process flow. If a customer was created, the user will see the customer number filled in if they went to the change customer transaction. Parameter IDs can be found by requesting help and technical details on a particular input field. Usually in the form of a 3-character code, Siamease allows this parameter id to be used as input by prefixing the PID with a %.

```
%MAT
```

If a task were to create a single material and the next task required that material number as input, the second task would have a variant that specified %MAT as the value of the material number. A PID will usually only be used and defined on a Change transaction.

SYSTEM VARIABLES $ $SY-FIELDNAME

System variables are set by the system at runtime. It is SAP's way of keeping track of what is happening and can be referenced by the variant for input. After a transaction has run a system message is usually presented to the user. The contents of that message are stored within system variables.

```
$SY-MSGV2
```

If a material was created using transaction MM01. After the material was saved the user would see the message: `MATERIAL 123 HAS BEEN CREATED SUCCESSFULLY`

Each part of the message would be available in the following fields.

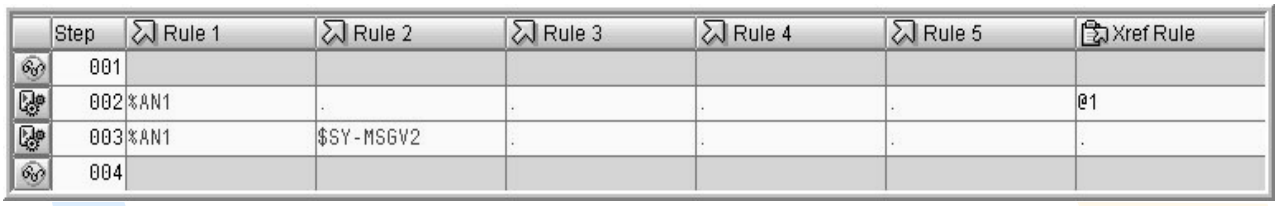
```
SY-MSGV1 = “MATERIAL”
SY-MSGV2 = “123”
SY-MSGV3 = “HAS BEEN CREATED SUCCESSFULLY”
```

STORE INPUT DATA @ @n

```
@12    - stores the 12th column / parameter loaded
```

This option refers to the column from the input data that you wish to save. If the data is loaded from a spreadsheet then @n refers to the column number to be saved. This option will also make any field conversions visible in the results. E.g. if an input field utilises an inbound conversion rule then the value after the conversion will be available in the results table.

ILLUSTRATED



	Step	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Xref Rule
	001						
	002	%AN1	.	.	.	.	@1
	003	%AN1	$SY-MSGV2	.	.	.	.
	004						

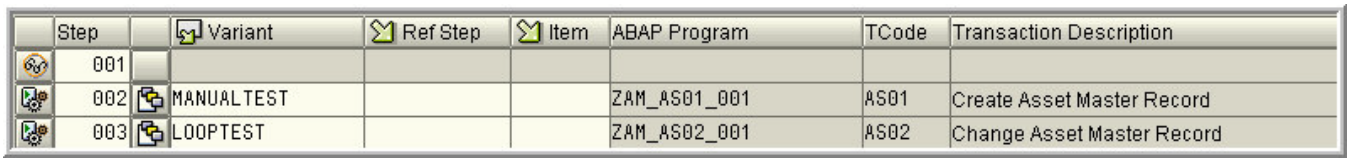
In the illustration above, Step 2 is configured to run 3 times. Each iteration will create an asset that is required in step 3. Step 3 has an inbound rule that references the old asset number in results-6 and returns the corresponding new asset number stored in results-1. The results of step 1 are stored in a results table illustrated below.

RESULTS TABLE								
RUN COUNT	STEP	ITERATION	RESULT 1	RESULT 2	RESULT 3	RESULT 4	RESULT 5	RESULT 6
1	002	1	100					Asset1
1	002	2	101					Asset2
1	002	3	102					Asset3
1	003	1	100	000000100				
1	003	2	101	000000101				
1	003	3	102	000000102				

INBOUND RULE

The inbound rules determine

- where the source of data is coming from
- how input fields vary their values
- how many records are to be processed
- what previous data is referenced



Step	Variant	Ref Step	Item	ABAP Program	TCode	Transaction Description
001						
002	MANUALTEST			ZAM_AS01_001	AS01	Create Asset Master Record
003	LOOPTEST			ZAM_AS02_001	AS02	Change Asset Master Record

Primarily a SAP variant defines the inbound rules. Data for each business scenario can either be referenced by a variant or can be captured explicitly into Manual Parameters against each specific task.

The script therefore needs to only refer to the inbound rules by specifying a relevant variant. The variant icon allows a variant to be configured for each task. When tasks are imported, the tasks default variant is imported as the default script variant.

DATA SOURCE

The data source defines from where data is being loaded and can be one of the following.

MANUAL	- Manual parameter values are captured and saved as a variant
UPLOAD	- data is read from a text data file that will be uploaded
CLIPBOARD	- data is imported from the windows clipboard
MEMORY	- data is received from another ABAP program (interfacing)

In addition to this, Scripting introduces another data source

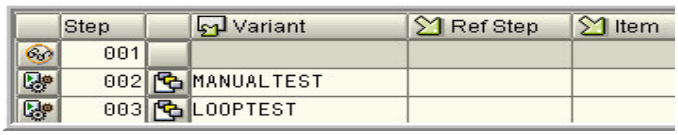
RESULTS	- data created by previous steps is made available in the form of 6 variables

HOW MANY RECORDS TO PROCESS

The **MANUAL data source** will only process one input record per Step iteration. The script can however repeat a step a number of times and thus create volume data. In each case the ITEM counter will increment and the LINE counter will reset to one on each call.

The **RESULTS data source** will refer to the last step that was executed and determine the number of records created by that. It will then copy the Manual Parameters input values for each result record. Each will then be processed individually. From this point on the task will behave as if it had uploaded a number of rows of data to be processed. The script counters will therefore increment the LINE counter whilst the ITEM counter remains constant. Since the script variables are incrementing, the manual parameter values can also vary provided that the INBOUND VARIANT COMMANDS are used (see below).

By default a step will refer to the result data created in the previous step. Other step results can be referenced either through inbound variant commands or by specifying the reference step or item on the Steps Inbound rules.



INBOUND VARIANT COMMANDS

The MANUAL data source is available to all Siamease tasks. Using normal SAP logic it allows a single set of scenario data to be stored by capturing the Manual Parameters and saving as a VARIANT. To extend standard SAP logic, Siamease provides **Inbound Variant Commands** that will allow formulae and script variables to be embedded into the manual parameters. Each time the task is run new values can now be substituted into the manual parameters fields.

&	Refers to the script variables and can be embedded anywhere in a field value
=	A formula where a string can be passed to the formula for processing
#	Results from previous runs can be accessed

VARIABLE SUBSTITUTION &

TEXT &C &S &I &L

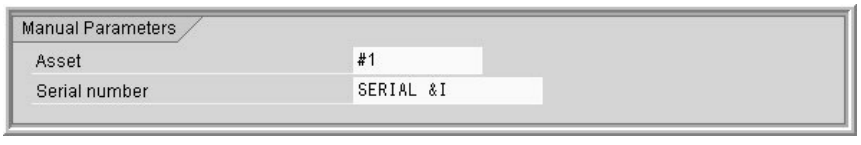
Any manual parameter value containing a &C, &S, &I, or &L will automatically be substituted with the current script variables. Each variable is a 3-digit number i.e. 001 to 999. The inbound variant command can be found anywhere in a string and is useful for varying descriptive information.

```
Serial number: [ 999&C&I ]
Customer name: [Joe Soap &I for class group &C ]
```

In the example above a step may be configured to repeat 4 times. Each iteration will increment the &I counter and will therefore results in creating 4 transactions each with a different serial number and customer name.

Will result in serial number 999001001 and Joe Soap 002 for class group 001…

```
Serial Number       Customer Name
999001001           Joe Soap 001 for class group 001
999001002           Joe Soap 002 for class group 001
999001003           Joe Soap 004 for class group 001
999001004           Joe Soap 005 for class group 001
```



RESULTS INPUT #1 - #6 #n C=counter S=step I=item L=line

Results Input is the key for linking steps together. The # command allows the result of a previous step to be captured into an input field on the next step. Each step has an outbound rule that determines how the results table will be populated. There are 6 rules that determine 6 results fields. The # command has access to these 6 results fields within the results table.

By default, the results of the previous step are referenced. This can be overridden by specifying a reference step against the step. This allows step 2 and 3 to refer to the results created in step 1.

If a particular field needs to deviate further then it can reference the results table right down to the individual line in the results table. If a step is processing multiple records then the # command will use the current row offset in the step to determine the results record unless otherwise instructed.

#1

A script may be setup to create a few materials, a few customers and then create sales orders using these same customers and materials. There are 3 tasks, one to create customers, one for materials and one for sales orders. The customer and material numbers are not known until they are created. The script would define outbound rules to ensure that the customer and material numbers were saved into the results table.

The script would then look as follows

```
STEP1: Create 1 customer (RULE #1 = customer number using $SY-MSGV2 )
STEP2: Create 2 materials (saving material number into result #1 and date into #2)
STEP3: add stock to material using #1

(Note that both materials will have stock added without explicitly stating material 1 or 2)

STEP4: Capture a sales order for customer      #1 S=001
Capture a line item referencing 1st material   #1 I=001 with reference step = 002
Capture a line item referencing 2nd material   #1 I=002 with reference step = 002
```

#n [“search string”]

This option searches the 6th results field and returns the results specified by #n .

```
#1 “oldno001”
```

When creating a customer master we may define 2 tasks, one to create the basic customer master and another to add additional bank details to a customer. This process was split into two because there are an unknown number of bank details for each customer. It makes sense to split into two tasks but creates the problem that the bank detail load does not know the customer numbers that are created in the first task.

The script is created to solve this problem. After executing task 1 (the basic customer master) the outbound rule saves the newly created internally assigned customer number. The outbound rule also specifies that the previous customer number must be stored in RESULTS using the @ command i.e. rule 6 equals @12 (discussed in the outbound variant commands). The results table will therefore contain the new customer number and the previous customer number.

Subsequent steps can then lookup results 6 and retrieve the new SAP number. In the example the second step would have defined a field in the upload spreadsheet that has the value #1 “980001”. Each line in the spreadsheet containing bank details can have a value referring to the relevant customer by means of its previous customer number.

FORMULA = =FM?text string

This option allows parameters to be passed to a Form routine that will process the input data and pass a result back. FM represents the form name to be called. All forms are stored in the Siamease ABAP program ZCSEZEFM that contains a suite of formulae that can be referenced.

Refer to VARIANT FORMULA to maintain.

When the task is being run in a script, script variables can also be embedded in the text string e.g. =CT?Some text with &C &S &L &I counters

Refer to SCRIPT VARIABLES.

```
=WORKINGDAYS?10
```

The above example calls a form that determines the date 10 working days from the current date. The parameter 10 is passes to the form called WORKINGDAYS. This means that the input parameter has defined a rule where the value will vary from day to day. Unfortunately this feature is only available to Siamease tasks.

ENHANCED VARIANT CONSIDERATIONS

ADDING SIAMEASE FORMULAE TO SPREADSHEET DATA

By default the Enhanced Variant functionality is only available to the sources MANUAL and RESULTS. If you wish to include these options in the other sources then the 'VE' option must be specified on the INCLUDE tab when generating the task. Specific functionality can also be added by add PI, SY or FM. By default these options are switched off when processing spreadsheets due to performance issues when no formulae is being utilised.

VARIANT FORMULA

Formulae are coded in program ZCSEZEFM. To access this program, execute the task builder (zeze), select the INCLUDE code tab and press the

button.

The example below is a typical formula.

- Value field is used to pass a value to the formula and to send a value back to the screen field.
- Return code is used to identify errors. If the formula were to detect an error (i.e. master data already exists) then a return code of 8 would abort the load of that particular row of data.

```
*---------------------------------------------------------
* FORMULA NWEEK
* The purpose of this form is to return next weeks date.
*---------------------------------------------------------
form NWEEK changing valuefield returncode.
  data: nextweek like sy-datum.
  clear returncode.
  nextweek = sy-datum + 7.
  write nextweek to valuefield.
endform.
```

Please note that program ZCSEZEFM must be transported to the live environment after any changes.

LINKING SCRIPTS

It may make business sense to separate certain data loads into different scripts. Although the results source is only available during the execution of a script, specific results can, however, be saved permanently and referenced to by other scripts at a later stage.

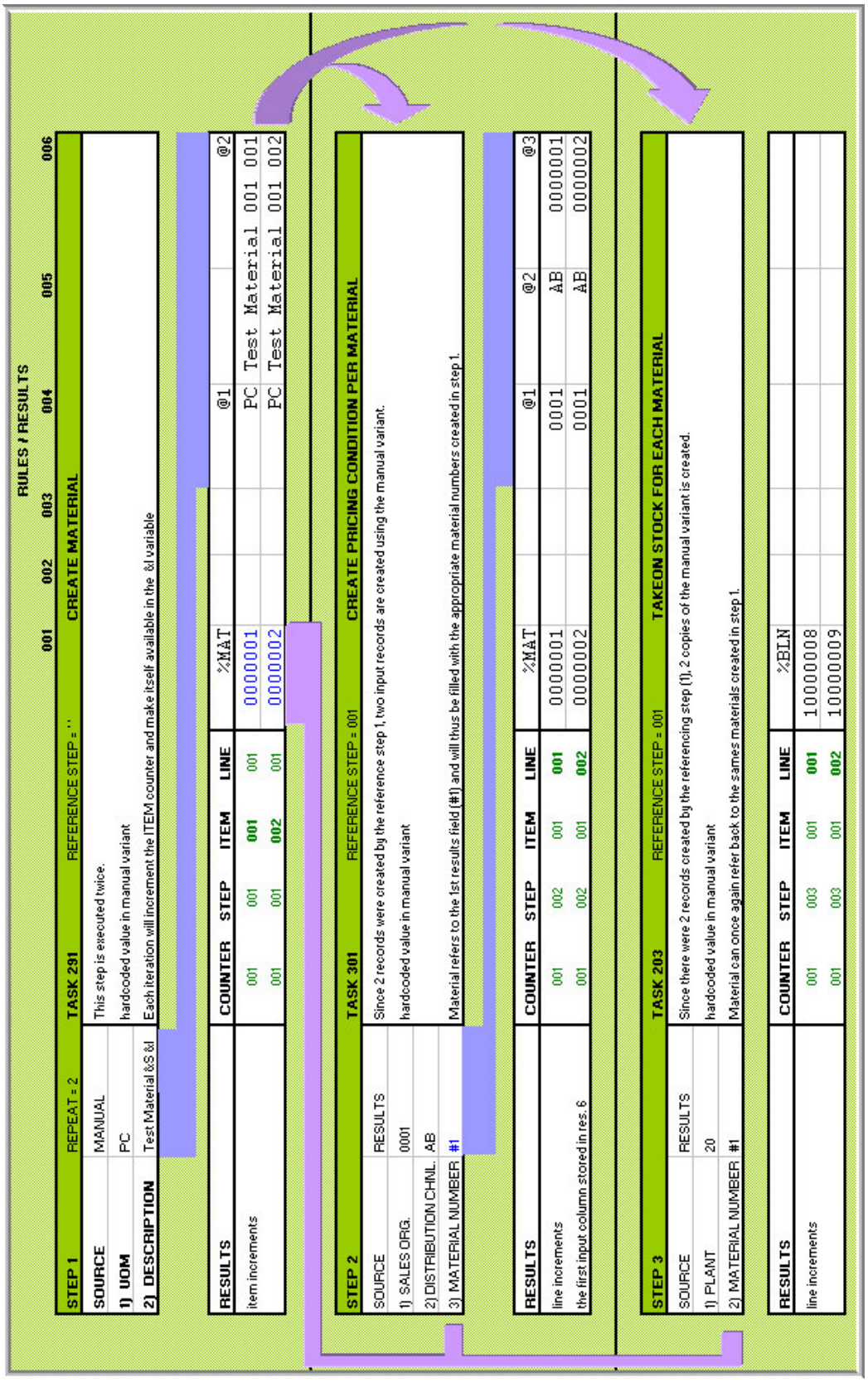
To save results data the specific step to be saved must have a SAVE name specified on the script. This data is then accessible to other scripts that specify a read name in the inbound rules section of the script.
When data is referenced via a save name the results are loaded into Step 000 for access. This data is then available to all steps that refer to RESULTS as the data source or those steps that utilise variant commands with the VE include option. If a step does not want to refer to step 000 when a read name has been specified then a reference step should be supplied to overrule the step determination.

Use table maintenance (sm30) on table zcsresults to delete or maintain results data created in error.

Please note that if multiple scripts use the same save name then resultant data is appended to the results table i.e. one scripts results do not replace another. E.g. Local customers loaded in one script in several steps, foreign customers loaded in another script. Each script can save the old and new customer number into the results table under the SAVE name CUSTOMERS. Another script can then reference ALL customers loaded by specifying CUSTOMERS as the read name.

Note that if a READ name is specified then RESULTS does not have to be the data source. Data can however be referenced using the # commands.

SCRIPT RULE SUMMARY


RULES / RESULTS
001 002 003 004 005 006
STEP 1 REPEAT = 2 TASK 291 REFERENCE STEP = '' CREATE MATERIAL
SOURCE MANUAL This step is executed twice.
1) UOM PC hardcoded value in manual variant
2) DESCRIPTION Test Material &S &I Each iteration will increment the ITEM counter and make itself available in the &I variable
RESULTS COUNTER STEP ITEM LINE %MAT @1 @2
item increments 001 001 001 001 0000001 PC Test Material 001 001
001 001 002 001 0000002 PC Test Material 001 002
STEP 2 TASK 301 REFERENCE STEP = 001 CREATE PRICING CONDITION PER MATERIAL
SOURCE RESULTS Since 2 records were created by the reference step 1, two input records are created using the manual variant.
1) SALES ORG. 0001 hardcoded value in manual variant
2) DISTRIBUTION CHNL AB
3) MATERIAL NUMBER #1 Material refers to the 1st results field (#1) and will thus be filled with the appropriate material numbers created in step 1.
RESULTS COUNTER STEP ITEM LINE %MAT @1 @2 @3
line increments 001 002 001 001 0000001 0001 AB 0000001
the first input column stored in res. 6 001 002 001 002 0000002 0001 AB 0000002
STEP 3 TASK 203 REFERENCE STEP = 001 TAKEON STOCK FOR EACH MATERIAL
SOURCE RESULTS Since there were 2 records created by the referencing step (1), 2 copies of the manual variant is created.
1) PLANT 20 hardcoded value in manual variant
2) MATERIAL NUMBER #1 Material can once again refer back to the sames materials created in step 1.
RESULTS COUNTER STEP ITEM LINE %BLN
line increments 001 003 001 001 10000008
001 003 001 002 10000009

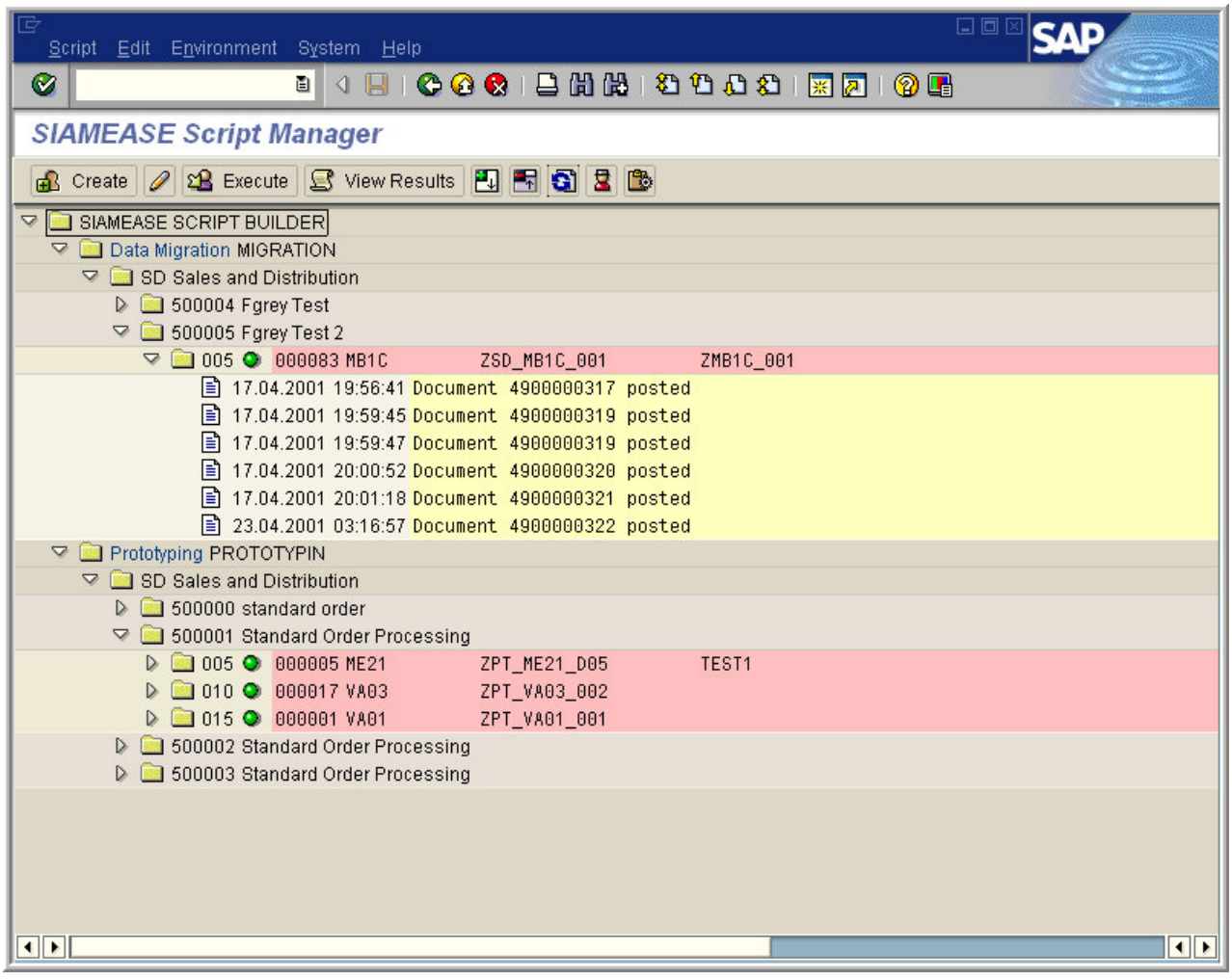
Chapter 7

SIAMEASE SCRIPT MANAGER ®

OVERVIEW

The Script manager provides control over Siamease scripts. Providing functionality to build, execute and maintain scripts. The Siamease Script Manager ® behaves in a similar fashion to the Siamease Task Manager ®. As with task, Scripts are classified according to projects and applications (not sub-application).

To execute the script manager press the icon (available on all Siamease screens or execute transaction code ZESB).


Script Edit Environment System Help
SAP
SIAMEASE Script Manager
Create Execute View Results
SIAMEASE SCRIPT BUILDER
Data Migration MIGRATION
SD Sales and Distribution
500004 Fgrey Test
500005 Fgrey Test 2
005 000083 MB1C ZSD_MB1C_001 ZMB1C_001
17.04.2001 19:56:41 Document 4900000317 posted
17.04.2001 19:59:45 Document 4900000319 posted
17.04.2001 19:59:47 Document 4900000319 posted
17.04.2001 20:00:52 Document 4900000320 posted
17.04.2001 20:01:18 Document 4900000321 posted
23.04.2001 03:16:57 Document 4900000322 posted
Prototyping PROTOTYPIN
SD Sales and Distribution
500000 standard order
500001 Standard Order Processing
005 000005 ME21 ZPT_ME21_D05 TEST1
010 000017 VA03 ZPT_VA03_002
015 000001 VA01 ZPT_VA01_001
500002 Standard Order Processing
500003 Standard Order Processing

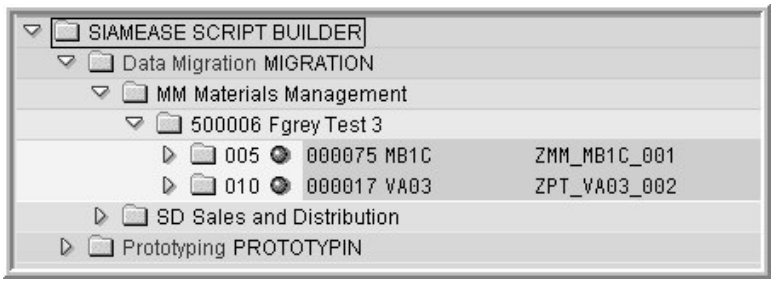
CREATING AND CHANGING SCRIPTS

CREATING NEW SCRIPTS

Press the create button to execute the Siamease Script Builder® - refer to Script builder for more information.

CHANGING EXISTING SCRIPT

Select a script to be maintained by expanding the tree and single clicking on the script line
Press the change detail icon in the application toolbar.
Siamease will execute the Script Builder and recall the script details.



EXECUTING SCRIPTS

Scripts can be executed via the Script Manager or the Script Builder. Press the execute icon.

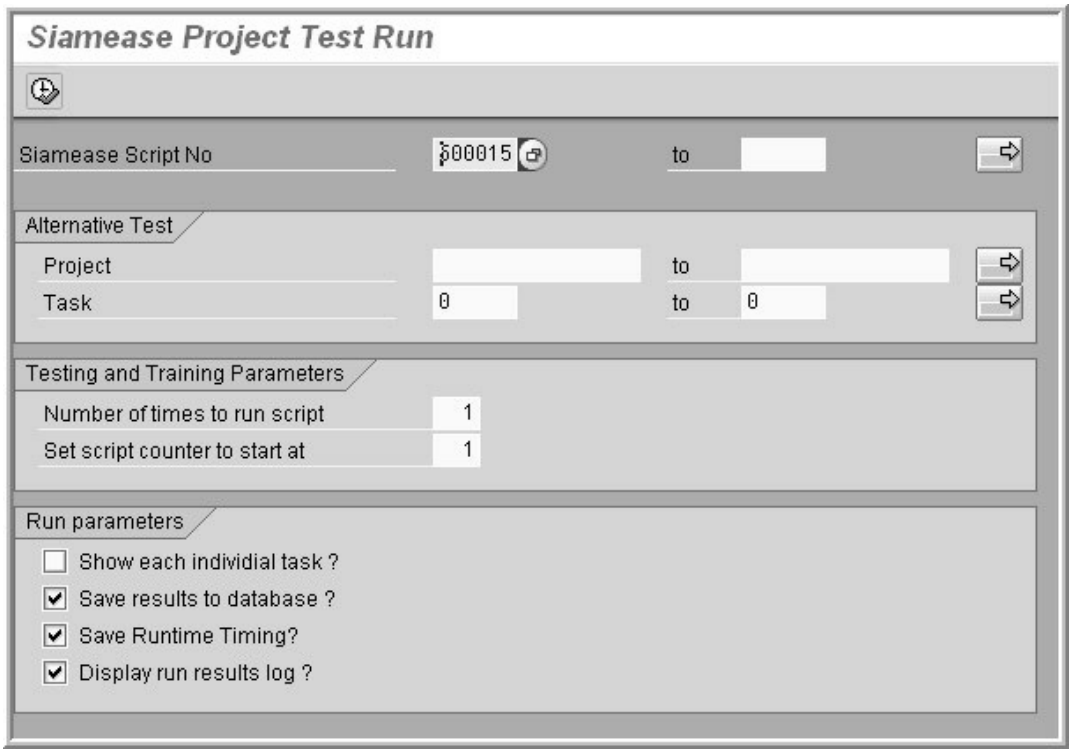
On the execution screen a counter can be set to repeat the entire script a number of times. A starting number can also be specified. The results of the run can be saved and are visible from the Script manager or by pressing VIEW RESULTS from the Script Builder. Results need only be stored if they need to be viewed at a later stage. Specifically useful for integration testing where previous test results are helpful for comparisons.

EXECUTE SCRIPT

To execute:

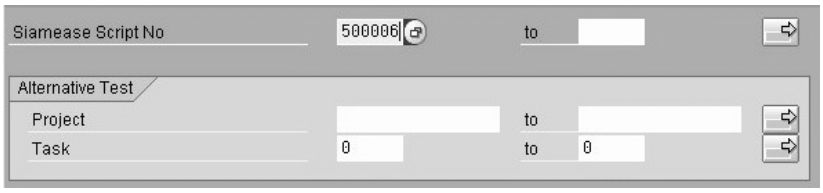
- Select the desired script (as above).

- Press the EXECUTE icon.



SELECTING TASKS / SCRIPTS

- The script executor can run a single script or many. It also has the ability to run selected tasks or all tasks within a project. Enter any combination of the following fields to select the desired task list.

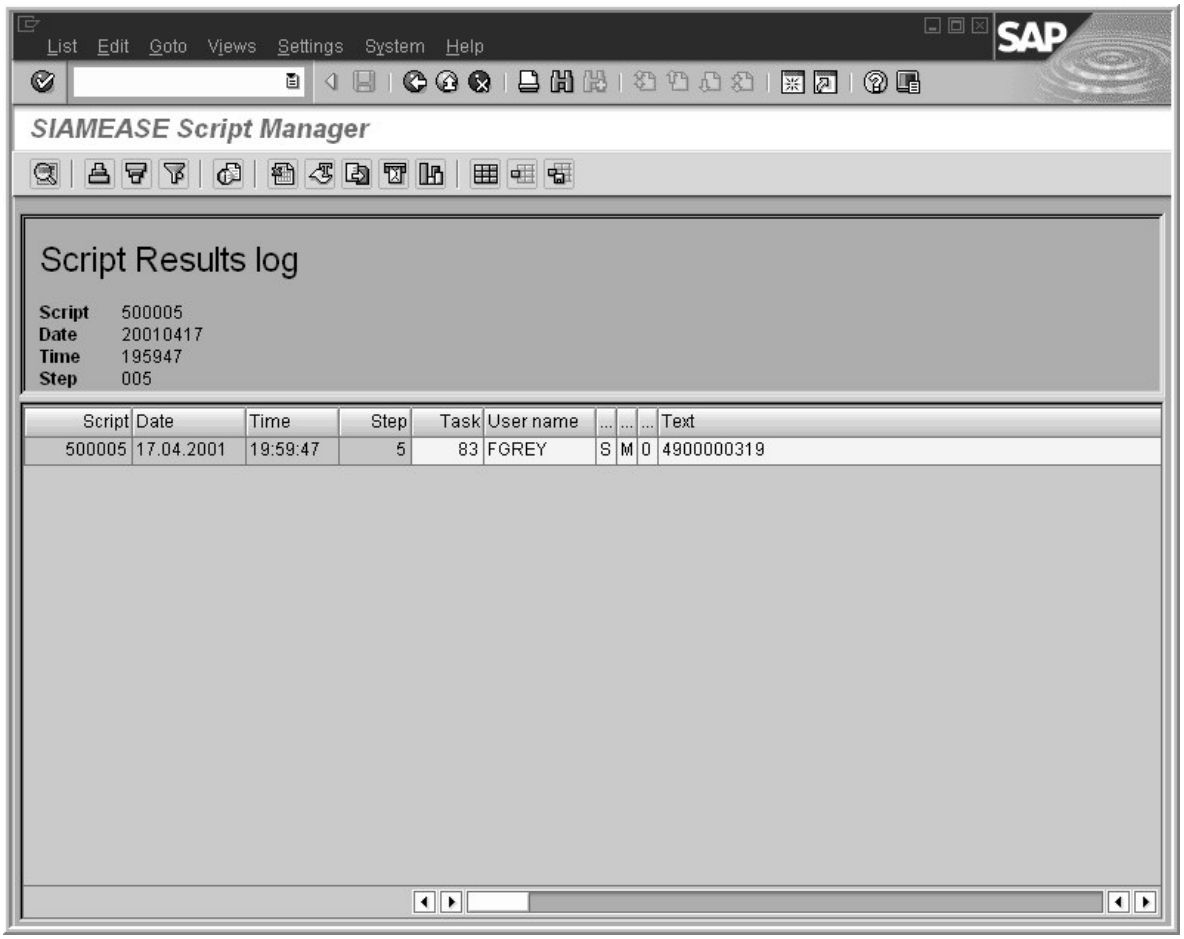


RUN PARAMETERS

- The run parameters allow the script to be executed more than once. By specify a high value a stress test can be initiated.
- If the {selection screens visible} option is checked then each task in the script once executed will display its own selection screen one after the other.
- Save results to database ensures that the test results are logged and will appear in the Script managers results log.
- For stress testing switch save results to database off, save runtime timing on and display run results on.

VIEW RUN RESULTS

Click on a script run result header (yellow line) and press the results icon. The results are displayed in a tabular format. The Script results log has functionality to create a spreadsheet or mail merge with results data. Select the appropriate function from the application toolbar.


List Edit Goto Views Settings System Help
SAP
SIAMEASE Script Manager
Script Results log
Script 500005
Date 20010417
Time 195947
Step 005
Script Date Time Step Task User name ... ... Text
500005 17.04.2001 19:59:47 5 83 FGREY S M 0 4900000319

Chapter 8

APPENDIX A – CASE STUDIES

THE TRAINING CLIENT

A Typical SAP implementation requires a Sandpit client where newly trained users can practice transactions learnt during the training course. The unfortunate reality is that this client will probably be filled with either the wrong data or with nothing at all. Even trainers who have prepared a training class are faced with the daunting task of creating prerequisite data in a client before training can commence. With skillful planning one training course can be planned before another so that the class has data, created by the previous class, to continue their business process training. Unfortunately training teams usually have very little time, having to prepare within the last hour.

The Siamease Scripting tool allows tasks defined for data migration to be reused for defining the training client. If defined correctly and completely a script would be quite capable of creating all the data required to train the class. Quite often the team that is training in one business module /application area will be dependant on another to prepare the data for their class (e.g. The Sales team may have to create customers, finished products, sales orders, deliveries and invoices before the Finance team can train around the financial aspects of Sales such as clearing open items against an incoming payment or processing an unpaid for an invoice already cleared).

Scripting allows a complete business process to be documented and implemented a number of times over. Each time the script runs it can create a free set of data that not only allows the trainer to prepare for training but also allows the playground client to be populated with clinically correct data that will allow new users to practice.

SCENARIO TESTING

Testing is often seen as a once off activity and in many cases projects learn the hard way that this is just not so. Business processes need to be documented and expected results needs to be stored so that future testing can be done quickly and efficiently. Usually manual scripts are created and testing is done manually, covering as many processes that can be manually achieved within the budgeted time frame. The downfall however is that in many cases the project hopes that everything will work, because the budgeted time frame usually does not cater for time spent on reengineering a business process.

The Siamease scripting tool allows business processes to be documented and implemented. Once defined completely and correctly integration testing can be done as simply as running a series of scripts. Any script that fails clearly identifies where the process has changed and where most of the time and effort must be applied. This results in more time spend on problem solving and allows more processes to be tested within the budgeted time frame. These tests can also be performed on a routine basis without any additional cost implication.

Results of previous tests can be compared against current tests and deviations can be detected and addressed.

STRESS TESTING

Stress testing is usually done manually or estimated based on the stress created during data migration. This often skews results due to the type of transaction posted but more than anything else is usually the Basis teams responsibility. Effective stress testing should be done with comprehensive business processes that simulate the real working environment.

Using the Siamese Scripting tool, scripts can be defined that replicate real business processes. Scripting allows input data to be generated and allows data to flow from one transaction to another. Scripts defined for data migration and / or defining training clients can be reused to process a sequence of transactions. Each script however can be run as many as 1000 times. Many different scripts can be run simultaneously and together a live working environment can be simulated. Buffers can be tested, queues monitored and testing can continue until the Basis Team are happy that instance and client parameters are set accurately.

END OF DOCUMENT